Declaration of Kevin Shank
Serial No. 09/223,972

EXHIBIT A

Nortel Server

Telephony Services Reference Manual

Document number: 203-7002-822
Product release: Release 1.1 Beta 2.1
Document release: Draft 01.02
Date:

NORTEL
NORTHERN TELECOM

5.0 Software architecture

This chapter describes in detail all software required for the Nortel Server product family. It provides an architectural overview and discusses implementation details.

The Nortel Server software architecture supports a variety of applications and services in a flexible, stable, and transparent manner. The use of an object-oriented distributed framework allows for the modular design of components within the system, and leverages existing assets.

The Nortel Server software architecture is based on open, well-specified components. This openness frees the system from the potentially tight binding between the architecture and particular hardware or software components.

5.1 Architectural overview

The Nortel Server software architecture is based on a distributed object model. The distributed object model supports a flexible, transparent, and stable system.

The distributed object model has the following characteristics:

- applications are composed of multiple distinct objects
- objects are encapsulated entities, which means that the interface to the object is the outside view of the object and establishes the set of operations that can be performed on the object
- objects have unique identities, allowing objects of the same type to be distinguished from one another
- objects can be located anywhere within the network, which means that invoking a method on an object that is located remotely is identical to invoking a method on a object that is in the same process
- threads can be used to provide increased responsiveness or to express concurrency

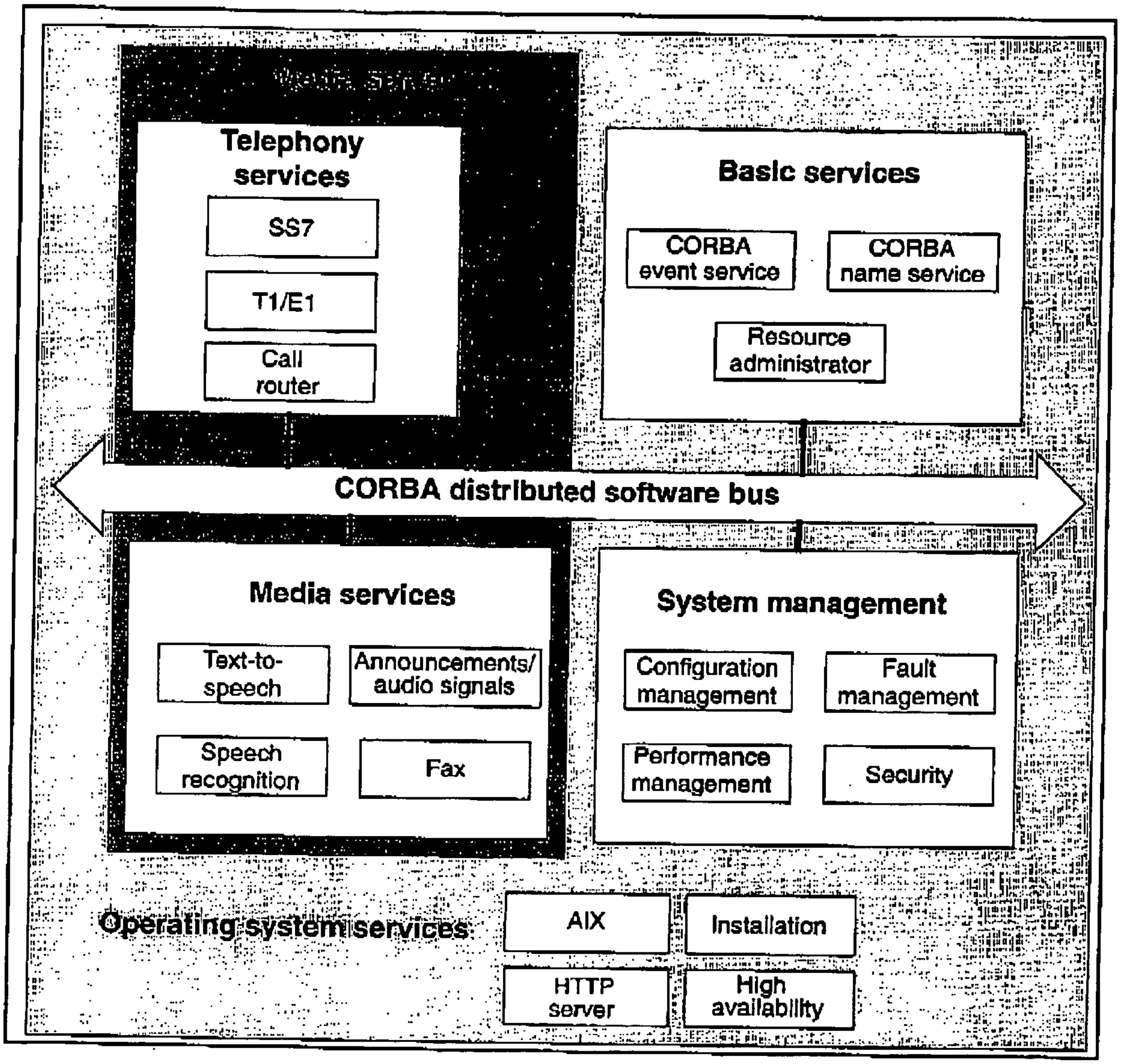
These attributes create a competitive architecture that supports rapidly changing applications.

The object model supported by Nortel Server is scalable from single node implementations to large scale systems comprised of multiple nodes. On a single node system, the software elements run in different processes on the same node. As systems grow, services and other system components can be located on various nodes throughout the system. In larger systems, services and system components can be replicated for greater availability.

Figure 5-1 shows a graphical representation of Nortel Server software architecture. Nortel Server architecture comprises the following subsections:

- Common Object Request Broker Architecture (CORBA) distributed software bus
- operating system services
- basic services
- media services
- telephony services
- system management

Figure 5-1
Software architecture overview



The following sections describe each subsection of the architecture.

5.1.1 CORBA distributed software bus

The Nortel Server software architecture uses an object-oriented communication framework. This framework provides location-transparent communication among the architectural components. The framework is provided by CORBA, which is an industry standard architecture developed by the Object Management Group (OMG) for systems integration.

A CORBA-based system is composed of co-operating objects on a software bus, which is called the object request broker (ORB). Each object has an interface, which in CORBA is an abstract, language-independent representation of the set of methods that can be understood by the object. Method invocations on remote objects occur through an underlying protocol, which can be specific to an ORB vendor or based on an industry standard.

CORBA 2.0 specifies a protocol that allows vendors of CORBA technology to interoperate. This means that objects implemented using one vendor's ORB can talk to an object using another vendor's ORB, assuming that both vendors adhere to the interoperability protocol.The developer is shielded from all details of the lower-level interaction, including the locations of the objects and the marshalling and unmarshalling of arguments.

Core interfaces

Nortel Server components, including hardware resources, such as voice channels, are uniformly viewed and treated as resources. Resources are controlled and managed through software resource wrappers. Resources and applications communicate with each other using the software bus.

From a software perspective, the following components are resources:

- media server
- resource administrator
- fault management
- performance management

The Nortel Server software architecture specifies a set of core interfaces that resources and applications can implement. The following core interfaces are available:

- management
- security/authentication
- recovery

At a minimum, resources must implement the recovery and management core interfaces.

Figure 5-2 illustrates resource interfaces.

Figure 5-2
Resource interfaces

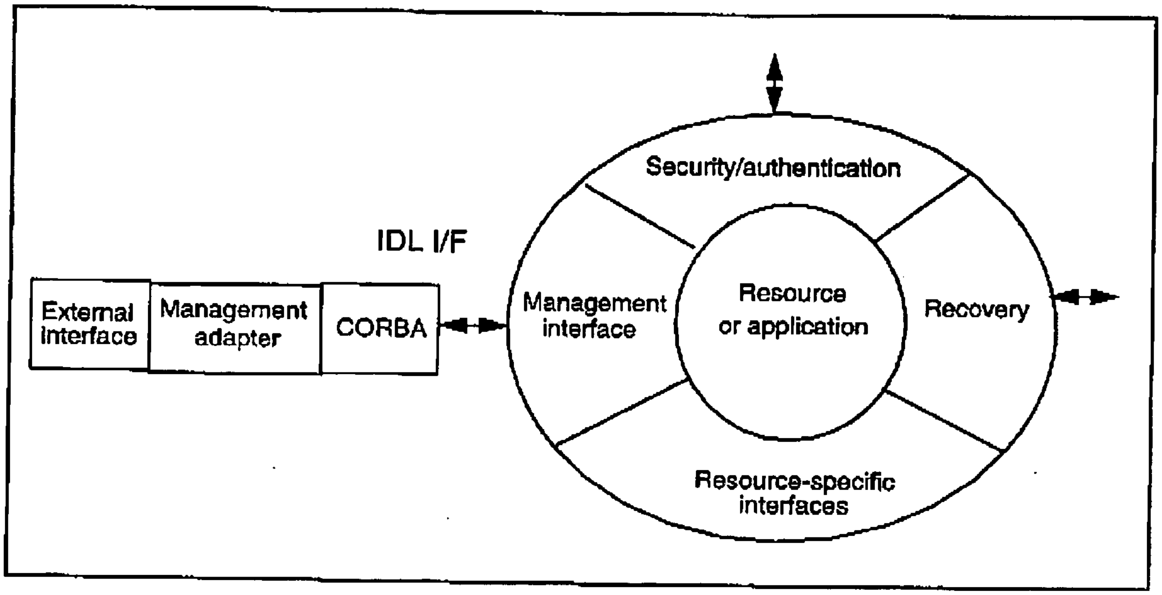


Table 5-1 describes core resource interfaces.

Table 5-1
Resource interfaces

Interface	Description
Management interface	Resources project an IDL interface that supports: • fault management, including a log and alarm interface • resource state management, which is based on the X.731 standard, including, — enable/disable — idle/active/busy — lock/unlock/shutdown • event notification which provides the ability for resources to register for specific events, and receive notification when the event is generated. For example, a resource can register for datafill changes. In addition, the management interface allows for custom adapters to be incorporated, which can be used to provide interfaces to legacy and standard systems.

Table 5-1
Resource Interfaces

Interface	Description
Security/authentication	The security/authentication interface interoperates with the core security/ authentication service to allow clients and servers to establish a mutual level of identity and trust.
Recovery	The resource recovery core interface cooperates with the resource administrator to make resources more available from a system perspective. It communicates with the resource administrator to handle the following situations: • resource failure • resource owner failure

5.1.2 Operating system services

Operating system services provide an underlying architecture that supports all Nortel Server functionality. Operating system services provide system-wide fucntionality that is applicable across all other platform services.

Operating system services are:

- the operating system
- high-availability (HA) services
- HTTP server
- installation services

Operating system

AIX is a robust UNIX operating system that is scalable and capable of supporting mission critical applications. AIX 4.1.5 is included in Release 1.0 Alpha 5.0 of Nortel Server, and is built on open standards, including UNIX95, XPG4, and X/Open.

High availability services

High-availability services provide active and stand-by 10BaseT or 100BaseT network interface cards (NIC) for applications that do not tolerate a single point of failure Ethernet LAN. NICs comprise Ethernet and FDDI adapters. Failure of a NIC or external hub is detected and logged, and network traffic is automatically switched to the stand-by NIC.

HTTP

As part of its basic services, the Nortel Server includes a standards-compliant HTTP server. The HTTP server, in conjunction with graphical clients, provides a window into the system.

Installation services

Installation services provide the ability to install a stand-alone Nortel Server or a line-up. Installation services allow the operations workstation to be installed, and additional machines to be installed from the operations workstation.

5.1.3 Basic services

This section provides a high-level view of the functionality provided by basic services, and explains how basic services interact at a high level with other architectural components in the Nortel Server software architecture.

Basic services are:

- CORBA name service
- CORBA event service
- resource administrator

CORBA name service

The name service allows the resource developer to place resources at specified locations, and allows other resources to find and access those resources by name.

CORBA event service

The event service allows objects to communicate in an event-driven style. The event service is based on three primary elements: suppliers, consumers, and event channels. Suppliers generate events, and consumers receive them. Event channels allow for indirect communication between suppliers and consumers.

The event service allows applications to determine which events are relevant to them and perform actions as required. The event service also allows for de-coupled design between generating and receiving events, which simplifies the implementation of object-oriented design and enhances scalability.

Resource administrator

The resource administrator provides network-wide tracking of resources. Each resource that becomes available registers with the resource administrator, providing information such as type and, optionally, any attributes or properties that distinguish it from other resources.

Resources can query the resource administrator for the existence of other resources by interface type and an optional list of properties. Resources are located by type rather than by name. In the case of several resources that match a particular type, the resource administrator uses a least-recently-used algorithm to determine which resource to use, and favors local resources over remote. This algorithm promotes load balancing of resource utilization.

The resource administrator tracks resource owners for resource recovery. This process reduces the amount of maintenance required to attain a high level of system availability. The tracking of resource owners can also be used for determining resource utilization.

The resource administrator also provides the first line of security. Resources cannot access other resources unless they have been authorized to do so.

The resource administrator is a multi-tiered service, which means it provides global resource tracking functionality. This includes local resource administrators that track nodal resources, and network-wide resource administrators that the nodal resource administrators can query if they cannot satisfy a request. This scheme is required for performance reasons and for high availability reasons. There is no single point of failure using the multitiered approach.

5.1.4 Media services

This section provides a high-level view of the functionality provided by media services, and explains how they interact at a high level with other architectural components in the Nortel Server software architecture.

Nortel Server's media services is provided by a component called the media server. The media server provides an object-oriented interface that is language independent and location independent. Applications using the interface can be collocated with the server, in separate processes, or even on different machines in a network. The media server interface is based on the ECTF S.100 C API.

The media server provides the following services for media interactions:

- player, which supports prompts and real-time text-to-speech
  - 24 and 32 kbps adaptive differential pulse code modulation (ADPCM) algorithms
  - 48 and 64 kbps mu-law and A-law pulse code modulation (PCM) algorithms
- recorder
  - 24 and 32 kbps ADPCM algorithms
  - 48 and 64 kbps mu-law and A-law PCM algorithms
- recognizer
  - natural language support
  - speaker-independent and speaker-trained
  - small to large vocabularies

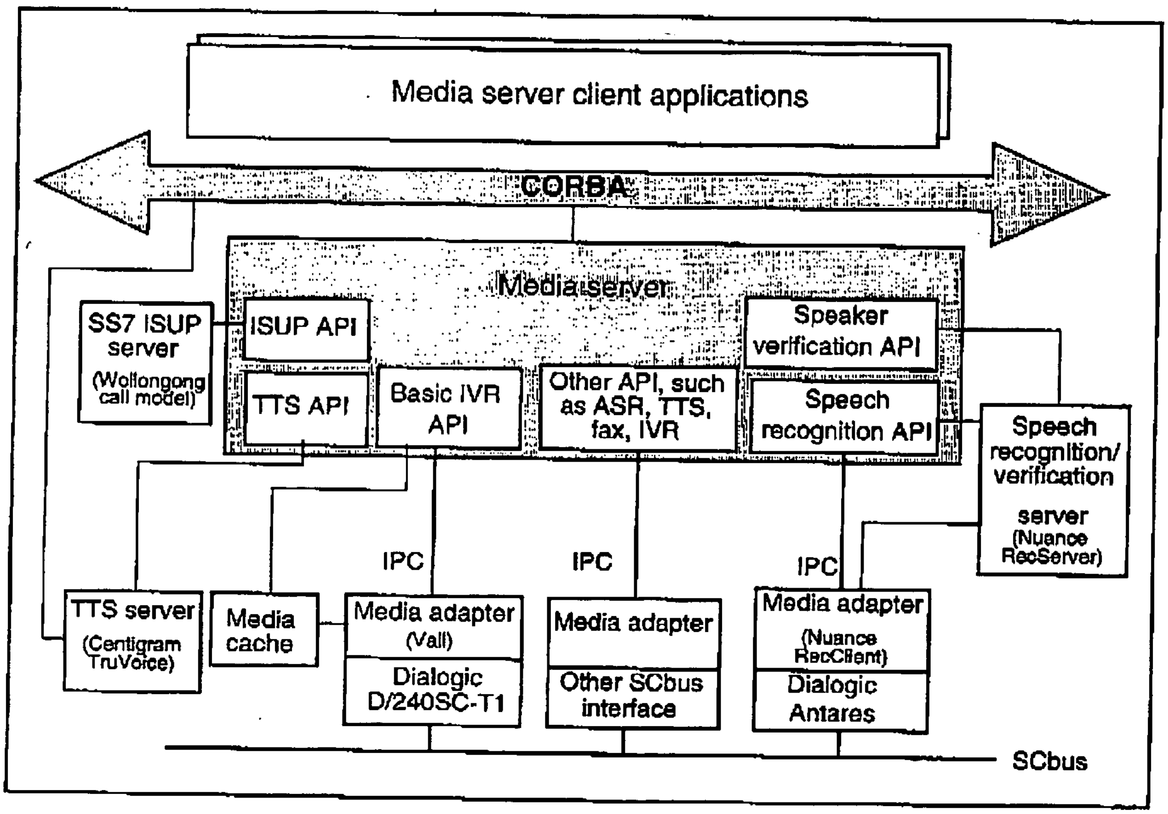
— multiple languages

- text-to-speech
- speaker verification
- signal generator
- signal detector

The media server provides an interface for bridging two parties using the time division multiplex (TDM) bus, which is the SCbus provided by Nortel Server media hardware.

Figure 5-3 shows the media server internal structure and interactions.

Figure 5-3
Media server internal structure and interactions



As shown in Figure 5-3, the media server accesses services through APIs provided by the various components. For example, Vail, a media adapter that presents a higher-level interface to the Dialogic D/240SC-T1 adapter, is accessed through the Basic IVR API. The Vail server resides in a separate

process. It is the responsibility of the API to transport method requests and responses, as well as asynchronous events between the media server and the Vail server. Other components have equivalent APIs that are used by the media server.

5.1.5 Telephony services

This section describes Nortel Server telephony services, and explains how they fit into the overall Nortel Server software architecture. Telephony services are part of the media server.

The media server uses the resource administrator to support multiple applications through the system call router. It obtains the voice and network ports from the resource administrator, which are then managed internally by the system call router. Applications that require access to network and voice ports use the call-control interfaces provided by the system call router.

Telephony services allow Nortel Server to interact with the telephone network. Telephony services are divided into two classes: call control services and transaction handling services.

Call control services include:

- making calls
- taking calls
- transferring calls
- joining calls

Transaction services include making SS7 Transaction Capabilities Application Part (TCAP) queries.

The following sections describe the architectural elements that provide telephony services:

- system call router
- T1/E1 signaling
- SS7 Integrated Services Digital Network (ISDN) User Part (ISUP) and TCAP

System call router

The system call router is a high-level interface that provides call control services. It is based on Enterprise Computer Telephony Forum (ECTF) S.100 standards, which are described at http://www.ectf.org, the ECTF web site.

It provides the following functionality:

- hides low-level call details, for example, whether signaling occurs in-band over the voice-channel or out-of-band through an external call-control protocol such as SS7 ISUP or primary rate interface (PRI)
- manages port types (incoming, outgoing, bi-directional)
- routes incoming calls based on port number or dialed number

T1/E1 basic signaling (robbed bit signaling)

Nortel Server supports the following types of in-band signaling over T1/E1:

- wink-start
- wink-start with digits, which is the same as wink-start, but the switch provides called number via dual-tone multifrequency (DTMF) after the wink)
- immediate-start
- clear-channel

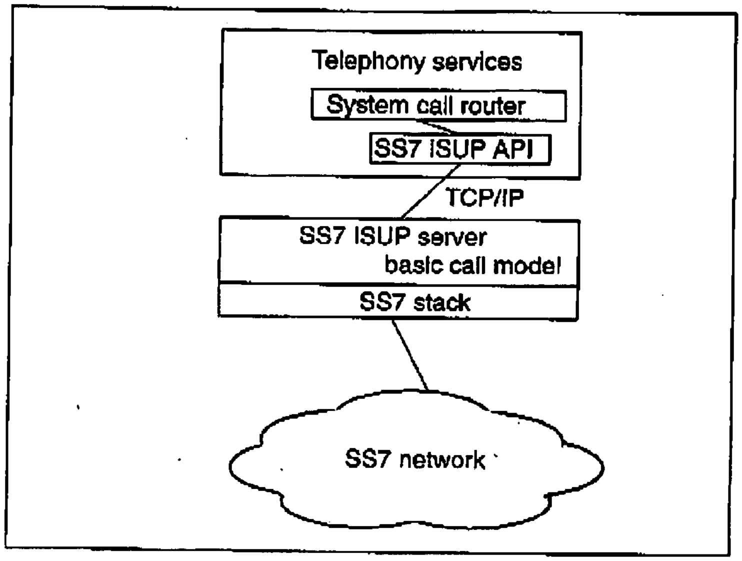
The system call router hides the details of the signaling type by providing a high-level call-control interface. The actual type of signaling used for each port is configurable.

SS7 ISUP

Calls originated and terminated using SS7 ISUP are handled in the same way as calls using in-band signaling. The system call router translates high-level call-control interfaces into SS7 ISUP messages using an SS7 ISUP application programming interface (API) that communicates with an SS7 ISUP server.

Figure 5-4 shows the architectural relationship between SS7 ISUP, the media server, and the SS7 network.

Figure 5-4
SS7 ISUP



An application can also access ISUP using an interface definition language (IDL) interface. This method of accessing ISUP is at a much lower level than most applications require. Accessing ISUP using an IDL interface provides the application with all of the ISUP messages, including the maintenance messages.

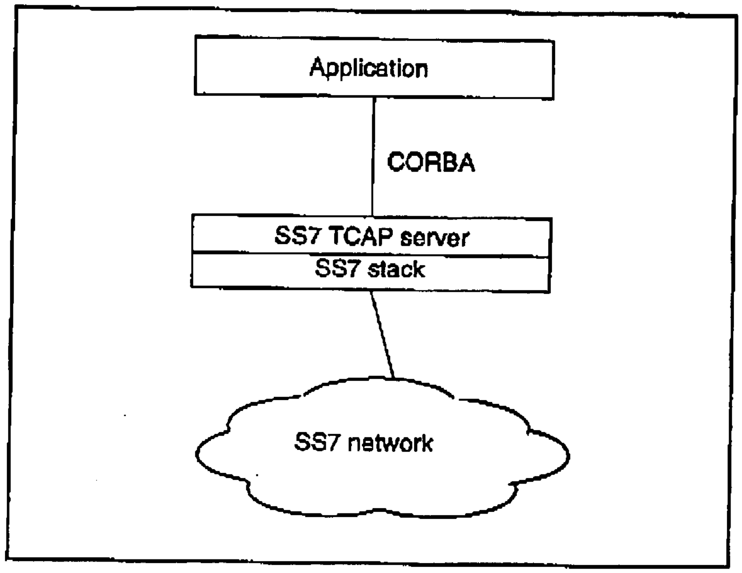
The SS7 ISUP server provides a basic call model which provides a number of functions including maintaining the state of calls, handling all maintenance messages, and providing APIs for enabling and disabling ports.

SS7 TCAP

SS7 TCAP is accessed by an application through IDL interfaces provided by Nortel Server. Translation between SS7 TCAP and IDL is provided by an SS7 TCAP server. This server functions as an adapter from the SS7 protocol to CORBA IDL.

Figure 5-5 shows the relationship between SS7 TCAP, the application, and the SS7 network.

Figure 5-5
SS7 TCAP



5.1.6 System management

System management consists of the following four categories, which are based on the Open Systems Interconnection (OSI) system management definition:

- fault management
- configuration management
- performance management
- security management

System and network management user interfaces use based on standard network and system management interfaces where possible. System management provides the APIs and UIs for applications, resources, network components, and hardware components to be managed in a consistent, and extensible fashion.

Fault management

Fault management provides a mechanism for generating, collecting, monitoring, and displaying fault events. Fault events are logs and alarms.

Fault events can be generated by many different sources including:

- an IDL API
- entries in a log file or the UNIX syslog mechanism

- AIX error log events
- Tivoli Monitors (including support for SNMP, TME10 Distributed Monitoring, TME10 Software Distribution)
- command line interfaces (CLI)

Fault management also provides an interface to a hardware-level alarm mechanism that supports external alarms on some hardware platforms. Hardware level interfaces consist of critical/major/minor/normal lamp indicators, Audible alarms, and dry relay contacts. The hardware-level alarm mechanism is available only on the Solano.

Configuration management

Configuration management provides the mechanisms for:

- system installation, which provides the capability to install the operations workstation, as well as any servers in a line-up
- software distribution, which provides the ability to define, administer, and distribute software and configuration data from the operations workstation
- resource state management, which allows applications and components within the Nortel Server environment to be managed from the operations workstation. State management is based on the ISO X.732 resource state model. Resource information includes:
  - — administration information, including enable and disable
  - — operational information, including lock, unlock, shutdown
  - — usage information, including idle, active, and busy
  - — status information, including alarm, shutting down, and initializing
  - — additional information
- configuration (datafill) management, which allows applications to define, generate, and distribute configuration information, as well as notify applications of configuration change events. A Java-based configuration tool is provided which allows designers to create graphical configuration templates for custom or legacy resources.

Performance management

Performance management provides a set of capabilities to monitor and measure key performance attributes of the system.

The performance measurement system allows applications to define, generate, collect, store, and display performance measurement information. Performance data is generated by applications using an IDL API. Generated

data is then collected on each server and sent to the operations workstation, where it is deposited in a Oracle database.

The performance monitoring system provides a mechanism to monitor key system or applications resources. Performance monitoring can support real-time and non-real-time viewing of the performance data. Non-real-time monitoring is provided by the Tivoli TME10 Distributed Monitoring product. Real-time monitoring is a optional component and uses the capabilities of the IBM Performance Toolbox/6000 product.

Security

Security management provides a GUI, based on the Tivoli TME10 User Administration product, to manage user accounts, passwords, expiration dates in a consistent easy to use interface. Security management integrates AIX security capabilities into fault management, which provides a mechanism to monitor and generate events based on AIX security violations.

5.2 Implementation details

This section describes implementation details for each architectural element, for example, whether the implementation is third-party or Nortel-developed.

5.2.1 CORBA distributed software bus

The CORBA distributed software bus used in Release 1.0 of Nortel Server is provided by Orbix 2.1.

Orbix consists of a CORBA version 2.0-compliant object request broker (ORB), IDL compiler, and related tools.

The ORB mediates between clients and implementations (of application objects) and must provide a standard interface to such clients, and another to such implementations. CORBA does not specify whether the ORB is a set of runtime libraries, a set of daemon processes, a server machine, or part of an operating system. An ORB can be any of these.

Orbix is implemented as a pair of libraries - one for client applications, and one for servers - and the orbixd activation daemon. orbixd need only be present at nodes running CORBA servers, and it is responsible for (re)launching server processes dynamically as required. Non-distributed client and server applications in the same process address space can be built using the server library alone. In this case, references from one object to another are direct (for example, they are direct pointers in C++ and operation invocations are simple C++ virtual function calls).

Because of its library implementation, the Orbix ORB is conceptually omnipresent: there is no distinct component which one can identify and state that this one component encapsulates the entire ORB. There is no central

component through which all object requests must pass: instead object requests are passed directly from the client code to the invoked object implementation.

The role of orbixd is to connect clients and servers for the first time. orbixd uses a simple database, the Implementation Repository, to obtain activation information for its servers; for each server the information includes the appropriate CORBA activation mode, the name of the associated executable image and any command-line parameters. An important component of Orbix is its compiler technology, which translates CORBA IDL into programming language code (for example C++) that performs remote calls. The generated code is sufficiently sophisticated so that programmers are not burdened with extra programming steps.

The ORB enables objects to make requests and receive responses transparently in a distributed environment. It is the foundation for building applications from distributed objects, and for interoperability between applications in hetero- and homogeneous environments.

The CORBA 2.0 standard specifies an interoperability protocol, Internet Inter-ORB Protocol (IIOP), which allows objects in different ORBs to communicate with each other. Through IIOP, a multi-ORB architecture is possible. Orbix can be used for general computing applications while RCP can be used for applications that require low latency and high throughput for remote method invocations. Other ORB products can interoperate with the Nortel Server ORBs as long as they support implementations of CORBA's standard Internet Inter-ORB Protocol (IIOP).

Figure 5-6 illustrates Orbix architecture.

Figure 5-6
Orbix architecture

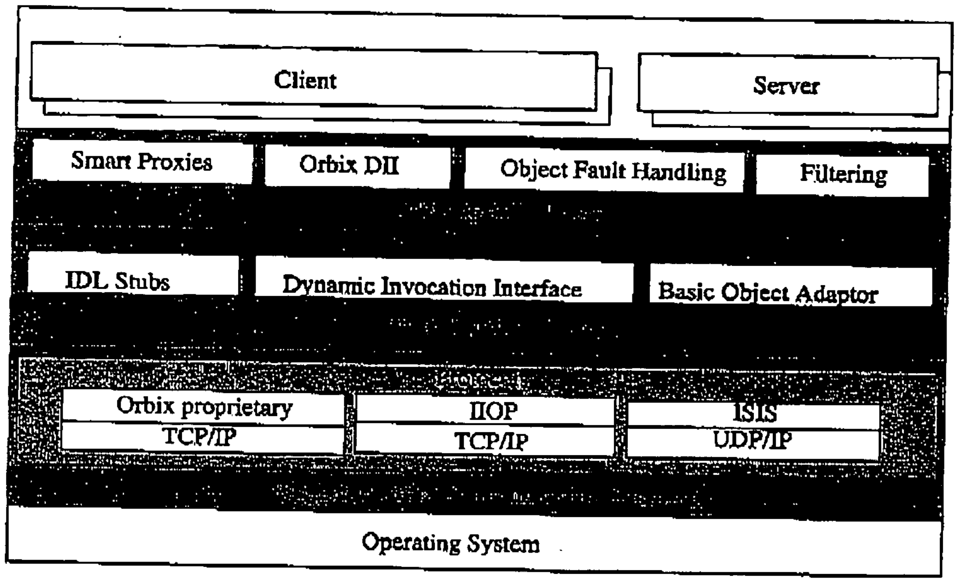


Table 5-2 describes Orbix architecture layers.

Table 5-2
Orbix architecture

Layer	Attributes
Orbix specific layer	• provides the following features that CORBA does not specify: — Smart Proxies, which allow the programmer to modify the behavior of the generated proxies, for example, to cache object state so that remote method invocation is not necessary when accessing state information — Filtering, which allows code to be executed before and after a method is invoked on a server object. Filtering can be applied per-object or per-class, and can be used to implement security policies between clients and servers, or to implement multi-threaded servers that create new threads for each incoming method invocation. — Orbix DII, which presents a stream-based interface of the CORBA dynamic invocation interface, enhancing the usability of DII for programmers — Object Fault Handler, which is used by Orbix's proprietary persistence mechanism to dynamically load server objects

Table 5-2
Orbix architecture

Layer	Attributes
Orbix runtime system	• implements CORBA client and server APIs • includes the following building blocks: — IDL stubs, which allows methods on remote objects to be invoked regardless of location and transport mechanism — Dynamic Invocation Interface (DII), which allows the programmer to dynamically create and dispatch method requests — Basic Object Adaptor, which provides basic object services, for example, object implementation start-up
Pluggable Orbix communication subsystem	• delivers messages between clients and servers • easy to substitute or extend to support other protocols • comprises the following protocols: — Orbix proprietary, which delivers messages between the client and the server — IIOP, which allows servers and clients in different ORBs to communicate with each other — ISIS, which enables high-availability through object replication — TCP/IP — UDP/IP
Operating system	• provides interfaces to the hardware

Orbix provides the name and event service that are part of Nortel Server basic services.

It also includes tools, such as an IDL compiler. The IDL compiler takes IDL specifications and generates the stubs and skeletons needed to implement client-side and server-side objects. The compiler handles the mechanical tasks of translating from a data format on one host to a data format on another host. It also handles generating code that translates method calls to network messages, and then back method calls. The process works similarly for return arguments.

Orbix also includes tools for manipulating the Implementation Repository. An Implementation Repository is contains a list of servers, and information about the servers, including their names, how they should be started, who has access to them, and when they should be started. These tools allow an administrator

to perform such functions as installing new servers, finding out what servers are in the repository, removing servers, and changing the owner of servers.

5.2.2 Operating system services

This section describes how operating system services are implemented for Release 1.0 of Nortel Server.

Operating system

AIX 4.1.5 is included in Release 1.0 of Nortel Server, and is built on open standards, including UNIX95, XPG4, and X/Open.

HA services

HA services are provided by Nortel-developed software.

HTTP

The HTTP server is provided through Netscape and supports web-based materials. Netscape's Enterprise World Wide Web Server creates, manages, and publishes information throughout an enterprise or across the Internet. The HTTP server is an open platform for developing and serving live, on-line applications using cross-platform tools based on the Java and JavaScript programming languages. It provides full-text and database search capabilities, management and control futures, and tools for creating and maintaining diverse forms of data.

Installation services

Installation services are provided by Nortel-developed software.

5.2.3 Basic services

This section describes how basic services are implemented for Release 1.0 of Nortel Server.

Name

The naming service for Nortel Server Release 1.0 is provided by OrbixNames. OrbixNames enables a programmer to associate meaningful names with object references within multiple, nested, and federated contexts. Additionally, OrbixNames supports the multi-threaded Orbix 2.01 MT as well as the initialization services module that is part of with Orbix 2.1.

Event

The event service for Nortel Server Release 1.0 is provided by OrbixTalk. It maintains a list of events that it routes to interested applications.

The event service is an optional ORB extension that allows clients to communicate with application objects using events. OrbixTalk is part of the events distributed services, and provides media services.

Resource administrator

The resource administrator is based on Orbix functionality. It is a Nortel-developed software component.

5.2.4 Media services

This section describes how media services are implemented for Release 1.0 of Nortel Server:

- Player functionality is provided as follows:
  - — Text-to-speech is provided by Centigram's TruVoice 5.1 and/or Eloquence Technologies Inc.'s Eloquent 2.0.
  - — Prompt playing is provided by the media server and Vail.
- Speaker verification is provided by Nuance.
- Recognizer functionality is provided by Nuance and VCS.

5.2.5 Telephony services

This section describes how telephony services are implemented for Release 1.0 of Nortel Server.

System call router

Release 1.0 includes a system call router that supports E1/T1 channel associated signalling and clear channel. It can route by called number or by port number.

SS7

SS7 connectivity in Release 1.0 of Nortel Server is provided by NewNet AccessMANAGER for SS7 signaling control and transaction handling using the IBM Artic 960 adapter. NewNet provides message transfer protocol (MTP) 1-3 and SCCP.

The ISUP server and the TCAP server are provided by Nortel development. The IDL adapters for ISUP and SCCP are also provided by Wollongong.

5.2.6 System management

System management implementation details are provided in Chapter 7, "System management."

Where possible, the infrastructure for system management is based on the Tivoli Management Environment (TME10), standard AIX utilities, and open standards. TME10 provides a framework for system management applications.